UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/798,916	03/11/2004	Scott J. Broussard	AUS920030818US1	7011

45502 7590 02/27/2009

DILLON & YUDELL LLP
8911 N. CAPITAL OF TEXAS HWY.,
SUITE 2110
AUSTIN, TX 78759

EXAMINER	
BROPHY, MATTHEW J	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
02/27/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/798,916	BROUSSARD, SCOTT J.	
	Examiner	Art Unit	
	MATTHEW J. BROPHY	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *15 December 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,7-15,17-25 and 27-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5,7-15,17-25 and 27-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _____ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to amendment field December 15, 2008.

2. Claims 1-5, 7-15, 17-25 and 27-30 are pending.

Response to Amendment

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-5, 7-9, 11-15, 17-19, 21-25, 27-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over as being anticipated by US PG Publication 2004/0078540 Cirne et al. hereinafter Cirne. In view of US Patent 7,089,460 Fu hereinafter Fu and further in US Patent 6,658,652 Alexander et al hereinafter Alexander.

Regarding Claims 1, 11 and 21 Cirne teaches: A method [system, or article of manufacture] for detecting memory leaks in a software program, said method comprising the steps of: monitoring a specified one or more analysis properties of software objects executing in the software program (**Cirne Paragraph "[0015] The present invention, roughly described, pertains to technology for identifying potential sources of memory leaks by tracking growth patterns of groups of stored items. One example of a group of stored items is an instance of a Java collection. If the growth pattern of a collection indicates that it may be the source**

of a memory leak, that collection is reported to a user and will continue to be tracked."), and identifying any software objects determined to have one or more analysis properties that exceeds that property's predetermined limit. **(Paragraph [0060] In step 322, it is determined whether the change counter is greater than the sensitivity counter. The sensitivity counter is a static number that corresponds to the sensitivity setting described above. For example, Table 2 shows that if the sensitivity setting is 10 then the sensitivity counter is 3, and if the sensitivity setting is 4 then the sensitivity counter is 7. Thus, the first time the first threshold is exceeded (e.g. where the threshold becomes 5.4); the change counter will equal 1, which is less than the sensitivity counter (step 332). Therefore, the collection is reported as not leaking in step 334. If the change counter is greater than the sensitivity counter, then the collection is reported as being a potential source of a leak in step 336. For example, if the sensitivity setting is 9, then the sensitivity setting will be 3. When the change counter is greater than 3, the collection will be reported as a potential source of a leak. In other words, when the size of the collection grows so that more than three thresholds have been exceeded, the collection is reported as being a potential source of a leak.").**

[wherein said statistics report includes] a notation for one or more software object classes that have instance counts that have grown by a factor of ten or more since a time period from when said software object classes are first reported [within said statistics report] **(Paragraph [0047] "A higher sensitivity value will report more collections as potential leaks. ...Based on the sensitivity value, the system**

determines a sensitivity counter value and a growth factor according to Table 2." See Table 2, by way of example, includes growth factors from 1 to 2 associated with the sensitivity values. Further See Paragraph [0061] "The lower the sensitivity counter, the more likely the collection will be reported as a potential source of a leak. … In general, the heuristics looks to see if the threshold value has been changed X times, where X is the value of the sensitivity counter. Once the threshold value has been changed X times, the collection is considered to be a potential source of a memory leak.") *[Cirne teaches that object collections over the growth factor threshold, which is set in ¶ 47 and Table 2, (in this embodiment the growth factors thresholds are between 1 and 2). In ¶ 61 Cirne teaches that any growth over the growth threshold is reported. Because the growth thresholds in this embodiment are between 1 and 2, any growth of a collection of a factor over the growth factor of 1.0-2.0 is reported (i.e. a "notation" is made), including growth by a factor of 10 or more. While this notation is not explicitly made in a "statistics report", the "statistic report" of the limitations is addressed below with the Alexander reference.]*

Cirne does not explicitly teach: wherein the one or more specified analysis properties includes one of an object's age; determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit

However, these limitations are taught by Fu: wherein the one or more specified analysis properties includes one of an object's age. **(Column 5, Lines 59-66, "FIG. 3 illustrates an exemplary cutoff weighting subroutine 300 that simply discards old memory usage elements. Memory usage weighting subroutine 300 begins at block 301 and proceeds to looping block 305 where an iteration through each usage data element begins. The first step in the loop is decision block 310 where a determination is made whether the current usage data element is older than a threshold time.")** determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit**(Column 5, Lines 59-66, "FIG. 3 illustrates an exemplary cutoff weighting subroutine 300 that simply discards old memory usage elements. Memory usage weighting subroutine 300 begins at block 301 and proceeds to looping block 305 where an iteration through each usage data element begins. The first step in the loop is decision block 310 where a determination is made whether the current usage data element is older than a threshold time.")**. In addition it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Cirne with the object age comparison of Fu as: Cirne teaches the detection of memory leaks based on suspicious characteristics of object groups, wherein one of the characteristics is allocation time (see Cirne e.g. Paragraph [0052]). Also, Cirne teaches the comparison of another characteristic to a threshold (group size, referenced below). Finally, one of ordinary skill in the art would be

motivated to combine the memory leak detection of Cirne with the object age threshold of Fu as the an object age above the threshold could be another indicator of a potential memory leak in the system of Cirne.

Cirne and Fu does not teach: generating a statistics report including the generated stack walkback for the at least one identified software object wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error and wherein the generated statistics report identifies the likely location of at least one memory_ leak in the software program.

generating a stack walkback for the at least one identified software object;

However, Alexander teaches:

generating a stack walkback for the at least one identified software object; **(Alexander Col. 17, Ln 38-53, "For example, at node 1152 in FIG. 11B, the call stack is CAB, and the statistics kept for this node are 2:3:4:1. Note that call stack CAB is first produced at time 2 in FIG. 10A, and is exited at time 3. Call stack CAB is produced again at time 4, and is exited at time 7. Thus, the first statistic indicates that this particular call stack, CAB, is produced twice in the trace. The second statistic indicates that call stack CAB exists for three units of time (at time 2, time 4, and time 6). The third statistic indicates the cumulative amount of time spent in call stack CAB and those call stacks invoked from call stack CAB (i.e., those call stacks having CAB as a prefix, in this case CABB). The cumulative time in the**

example shown in FIG. 11B is four units of time. Finally, the recursion depth of call stack CAB is one, as none of the three routines present in the call stack have been recursively entered.") and

generating a statistics report including the generated stack walkback for the at least one identified software object… wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error and wherein the generated statistics report identifies the like of location of at least one memory leak in the software program. **(Alexander III Col 18, Ln 12-34"Tracing may also be used to track memory allocation and deallocation. Every time a routine creates an object, a trace record could be generated. The tree structure of the present invention would then be used to efficiently store and retrieve information regarding memory allocation. Each node would represent the number of method calls, the amount of memory allocated within a method, the amount of memory allocated by methods called by the method, and the number of methods above this instance (i.e., the measure of recursion). Those skilled in the art will appreciate that the tree structure of the present invention may be used to represent a variety of performance data in a manner which is very compact, and allows a wide variety of performance queries to be performed.")**

In addition it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Cirne with the metrics of Alexander as Alexander allows the developer to analyze the statistics and trace record to debug a memory leak

because “from such a report an analyst may notice an unexpected number of live objects of a particular class.” (Alexander Col. 39, Ln 45-47.)

Regarding Claims 2, 12 and 22, Cirne teaches: The limitations of claims 1, 11 and 21, further comprising the step of calculating an object's age by timing a current period starting when the respective object was instantiated **(Paragraph [0053] “The entry in the log file created in step 266 includes the following information: current timestamp when written to the log, an identification (ID) for the collection, the class of the collection, the allocation time of the collection, allocation stack trace for the collection, current size of the collection and ten sample elements in the collection (represented by class name, followed by the toString( ) representation capped at 20 characters).”)**.

Regarding Claims 3, 13 and 23, Cirne teaches: The limitations of claims 1, 11 and 21, further comprising the step of calculating object instance count growth as the magnitude of growth of an object's instance count over a given time period **(Paragraph [0051] “At the time for repeating the process, Agent 8 first checks the size of each collection (step 262). Each collection stores its own size. Agent 8 will sweep through all of its weak references and make sure that each object is still present in the heap by performing a simple null check. For each object that is still there, Agent 8 will read the size of that collection. In step 264, Agent 8 will update the heuristics for each collection for which it received a size. The heuristics (to be**

described in more detail below) determines whether the collection is a potential source of a leak or not.").

Regarding Claims 4, 14 and 24, Cirne teaches: The limitations of claims 1, 11 and 21, wherein the step of monitoring comprises monitoring objects within a class designated for monitoring **(Paragraph [0018] "One implementation of the present invention includes a method of monitoring for potential stores of memory leaks. The method includes tracking the size of a first group of stored items and determining whether that first group of stored items is a potential memory leak source based on change in size of the first group of stored items.")**.

Regarding Claims 5, 15 and 25, Cirne teaches: The limitations of claims 1, 11 and 21, further comprising the step of performing a stack walkback for the identified software objects **(Paragraph [0049] "If leak detection is enabled and the time out period has not expired (step 206), then the code in the constructor for the collection object will create a stack trace for the collection object in step 208. In step 210, the code in the constructor for the collection object will pass a reference to the collection object and the stack trace to Agent 8.").**

Regarding Claims 7, 17 and 27, Cirne teaches: The limitations of claims 6, 16 and 26, further comprising the step of generating a statistics report comprising the identified software objects **(Paragraph [0016] "In one embodiment, the present invention includes looking for collections that appear to be growing in size. These collections are flagged as potential sources of leaks. The system then reports information for these collections as metric data as well as to a log file. If a**

flagged collection no longer appears to be leaking, that change in status will be reported; however, the system will continue tracking and reporting data for that collection.").

Regarding Claims 8, 18 and 28, Cirne teaches: The limitations of claims 6, 16 and 26, further comprising the step of generating a web interface for user viewing of the statistics report at a computer display **(Paragraph [0032] "The workstations (e.g. 124 and 126) are the graphical user interface for viewing performance data.")**.

Regarding Claims 9, 19 and 29, Cirne teaches: The limitations of claims 1, 11 and 21, wherein the software objects are Java objects **(Cirne Paragraph "[0015] The present invention, roughly described, pertains to technology for identifying potential sources of memory leaks by tracking growth patterns of groups of stored items. One example of a group of stored items is an instance of a Java collection. If the growth pattern of a collection indicates that it may be the source of a memory leak, that collection is reported to a user and will continue to be tracked.")**.

3. Claims 10, 20 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over as being anticipated by US PG Publication 2004/0078540 Cirne et al. hereinafter Cirne. In view of US Patent 7,089,460 Fu hereinafter Fu and further in US Patent

6,658,652 Alexander et al hereinafter Alexander and further in view of US Patent 6,189,141 Benitez.

Regarding Claims 10, 20 and 30, Cirne teaches:

In addition Cirne further teaches: monitoring an amount of available memory for a software program referencing software objects **(Paragraph [0035] "A metric is a measurement of a specific application activity. Probes can be used to enable the reporting of a set of metrics for a managed application. Examples of metrics collected can include CORBA method timers, remote method indication method timers, thread counters, network bandwidth, JDBC update inquiry timers, servlet timers, Java Server Pages (JSP) timers, system logs, file system input and output bandwidth meters, availability and used memory, enterprise Java bean times, etc.")**; and upon such determination, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for a software program referencing software objects dropping below an amount of available memory necessary to store a current stack walkback **(Paragraph [0049] "If leak detection is enabled and the time out period has not expired (step 206), then the code in the constructor for the collection object will create a stack trace for the collection object in step 208. In step 210, the code in the constructor for the collection object will pass a reference to the collection object and the stack trace to Agent 8." While the determination is not taught by Cirne as explained below, the storing of the current stack trace is inherently prior to reaching the threshold as Cirne allocates memory for the trace).**

Cirne does not explicitly teach: determining when the amount of available memory for the software program referencing software objects is within a predetermined threshold amount of memory within zero memory available for the software program utilizing software objects; upon ~~such determination~~ determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software storing a current stack walkback of currently referenced software objects prior to the amount of available memory for [[a]] the software program referencing software objects dropping below an amount of available memory necessary to store [[a]] the current stack walkback.

However, Benitez teaches: determining when the amount of available memory for the software program referencing software objects is within a predetermined threshold amount of memory within zero memory available for the software program utilizing software objects **(Benitez Col 38 Ln 66-Col 39, Ln5 "It is now assumed for illustrative purposes that storage locator 1210 has set the overflow flag. If the amount of memory made available by the elimination of cold traces, as described above, has been sufficient to reduce the amount of memory used in hot trace storage area 203 below the overflow threshold, cold trace detector and remover 1220 need not further identify cold traces for removal.")** upon ~~such determination~~ determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software storing a current stack

walkback of currently referenced software objects prior to the amount of available memory for [[a]] the software program referencing software objects dropping below an amount of available memory necessary to store [[a]] the current stack walkback. **(Benitez Col 38 Ln 66-Col 39, Ln5 "It is now assumed for illustrative purposes that storage locator 1210 has set the overflow flag. If the amount of memory made available by the elimination of cold traces, as described above, has been sufficient to reduce the amount of memory used in hot trace storage area 203 below the overflow threshold, cold trace detector and remover 1220 need not further identify cold traces for removal.").**

In addition it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Benitez as Benitez use of thresholds allow the system to avoid memory overflows.

Response to Arguments

4. Applicant's arguments filed December 15, 2008 have been fully considered but they are not persuasive.

In remarks, Applicant Argues:

Cirne teaches that an operator of a system is able to set various configuration values in a configuration file. The configuration file includes a sensitivity value that determines how sensitive the tool will be to potential leaks. Based on the sensitivity value given, the system determines a sensitivity counter value and a growth factor. (See Cirne, ¶[0047]

and Table 2). However, Cirne merely describes arbitrary growth factor values when comparing the size of a collection representing a group of objects (called a "collection"): …Thus, the teachings in Cirne suggest that there is an arbitrarily-set, sliding scale of thresholds based on the sensitivity setting that is arbitrarily set to the system (See Cirne, Table 2). Moreover, Cirne does not disclose that its log file (assuming alog file were equivalent to Applicants' statistics report, which Applicant submits it does not) includes a notation for at least one software object class that has an instance count that has grown by a factor often or more since a time period from when the software object classes are first reported within the statistics report, as recited in amended claims 1, 11, and 21.

Examiner's Response:

Examiner respectfully disagrees. As described above, Cirne sets a growth factor threshold, over which growth is reported. (¶¶ 47, 61) In a specific embodiment the threshold is between 1.0 and 2.0 (Table 2). Because any growth over ten is necessarily over 2.0, Cirne would report (i.e. include a notation) this growth. While Cirne does not explicitly teach that the reporting is done in a "statistics report", the Statistics report is generated by Alexander, and provides a vechile for statistics useful for memory leak debugging that would be obvious to one of ordinary skill in the art at the time of the invention to combine with the growth factor reporting in Cirne because "from such a report an analyst may notice an unexpected number of live objects of a particular class." (Alexander Col. 39, Ln 45-47.)

Applicant's arguments concerning Fu and Alexander regarding the "notation" are moot in view of the examiner's reliance on Cirne. here.

In Remarks, Applicant Argues:

The Examiner has cited Benitez to assert that the reference teaches the above cited limitation [of claim 10]. However, Applicant respectfully disagrees. Benitez teaches that when all overflow condition is present (i.e., a hot trace storage area is becoming full), cold traces within the hot trace storage area are removed to make storage space for additional hot traces; see col. 35, lines 29-40 of Benitez). Benitez does not teach storing a current stack walkback when it is determined that the amount of available memory is within a threshold, as recited in claims 10, 20 and 30. Thus, in Benitez, the cold traces that were previously stored in the hot storage area are not stored. Rather, the cold traces are removed altogether, according to Benitez.

Examiner's Response:

Examiner respectfully disagrees. Benetiz teaches "upon determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software storing a current stack walkback of currently referenced software objects prior to the amount of available memory for [[a]] the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback" because: Benetiz teaches

"determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available" by determining when an overflow condition exists and "storing a current stack walkback of currently referenced software objects prior to the amount of available memory for [[a]] the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback" by Storing Hot traces in the hot trace area until the overflow happens. (Benitez Col 38, Ln 66-Col 39, Ln 5).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to MATTHEW J. BROPHY whose telephone number is 571-270-1642. The examiner can normally be reached on Monday-Thursday 8:00AM-5:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MJB

2/24/2009
/Wei Y Zhen/
Supervisory Patent Examiner, Art Unit 2191